Method and Apparatus For Improving File System Proxy Performance and Security By Distributing Information To Clients Via File Handles

Background of the Invention

Technical Field

The invention relates to file systems. More particularly, the invention relates to a method and apparatus for improving file system proxy performance and security by distributing information to clients via file handles.

Description of the Prior Art

The Network File System (NFS), developed by Sun Microsystems, is the *de facto* standard for file sharing among UN*X hosts. NFS Version 3 is documented in RFC 1813. NFS is a stateless protocol. This means that the file server stores no per-client information, and there are no NFS connections. For example, NFS has no operation to open a file because this would require the server to store state information, *e.g.* that a file is open, what its file descriptor is, the next byte to read, *etc.* Instead, NFS supports a lookup procedure, which converts a filename into a file handle. This file handle is a unique, immutable identifier, usually an i-node number, or disk block address. NFS does have a read procedure, but the client must specify a file handle and starting offset for every call to read. Two identical calls to read yield

the exact same results. If the client wants to read further in the file, it must call read with a larger offset.

A software program or appliance that is a proxy for the NFS protocol, or any other protocol that uses server-generated file handles, usually requires additional file metadata information to be stored either on the server or locally on the proxy, especially in the case of encrypting or authenticating client data, and also in the case of server virtualization, *i.e.* when serving as a single access point for clients, while providing them with access to several servers on the back end. This metadata can be used, for example, to apply different encryption keys, or to enforce access restrictions to files that are located in different logical units that are defined on the proxy, but possibly invisible to the file server.

Storage encryption appliances that secure data on NFS file servers are an example of a device that matches the above characteristics. Such an appliance forwards file handles generated by the file server to clients, and subsequently acts as a proxy for client requests for access to the file system on the server. Given a file handle from a client, the appliance needs to establish to what area (a.k.a. storage vault) the file belongs, and use the appropriate keys to encrypt or decrypt data. If the metadata used to establish this are not available on the proxy, as is typically the case with large file sets accessed by many client machines, the proxy must send additional requests to the file server to determine how to handle the client request correctly.

It would be advantageous to provide a mechanism that distributes, and effectively caches, information by inserting it into file handles that the proxy sends to clients. It would also be advantageous to provide a mechanism that improves performance by

eliminating the need for the proxy to generate additional requests to the server to establish file identity.

Summary of the Invention

The preferred embodiment of the invention distributes, and effectively caches, information by inserting it into file handles that the proxy sends to clients. This information can be used to improve performance by eliminating the need for the proxy to generate additional requests to the server to establish file identity. The distributed information can also be intended to improve security, for example, by allowing the proxy to encode into the file handle a session key that expires after some amount of time.

Brief Description of the Drawings

Figure 1 is a block schematic diagram that shows how a proxy appliance or software program can distribute information in client file handles according to the invention;

Figure 2 is a block schematic diagram that shows how a proxy appliance or software program uses the information in client file handles when it receives a request from a client according to the invention; and

Figure 3 is block schematic diagram that shows the additional operations that must be executed if the information is missing from the file handle.

Detailed Description of the Invention

A proxy that is logically, *i.e.* in terms of data flow, located between clients and servers in general must remember certain information about the handles, or generally object identifiers, issued by a specific server to a specific client. For example, if the proxy serves more than one server, the proxy must remember which server issued a specific handle. This is necessary because the server name is not necessarily mentioned in future access by the client,

One approach is to keep a table inside the proxy. One disadvantage of doing so is the size of the table and the fact that this table might be lost if the proxy is rebooted. Saving such a table in a battery-backed RAM or some other non-volatile storage is possible, but expensive in terms of performance, cost, or both.

The preferred embodiment of the invention modifies the handle returned to the client. When the client asks the server to issue a handle, the handle is intercepted, and instead of returning the handles to the client as-is, additional information is encoded into the handle. Examples of such additional information include server name/ID, time, temporary session key, other cryptographic information, etc. One advantage of such application is that there is no need to keep a table in the proxy.

The preferred embodiment of the invention distributes, and effectively caches, information by inserting it into file handles that the proxy sends to clients. Information can be inserted into file handles in any of various ways. If the file handles are of a variable size, then additional bytes can be inserted at the beginning, at the end, or in any other possible way, and the size field can be updated to reflect the larger size. If file handles have a fixed size, there are other approaches that can be used. For

example, some file servers do not use the full size of file handles and in this case the proxy can use the available space to store information before sending the file handles to clients.

This information can be used to improve performance by eliminating the need for the proxy to generate additional requests to the server to establish file identity. Performance improvements can be achieved by storing information, such as a key ID or some other type of metadata, in the file handle, and thus avoiding having to send additional requests to the file server when the file handle is used. The distributed information can also be intended to improve security, for example, by allowing the proxy to encode into the file handle a session key that expires after some amount of time, and/or to sign the file handle contents so that clients can not tamper with it. By implementing a session key in file handles, it is possible to require that at certain intervals, for example intervals in time or in usage, clients establish a session by using a client-side application that requires stronger authentication than that allowed by the plain storage protocol. For instance, the typical method of authenticating clients in the NFS protocol is to trust the user ID (UID) sent by the client machine, which is a very weak form of authentication. A more advanced form of authentication can be enforced by using an alternative program that mounts NFS exports and then embeds a session ID into the root file handles of those exports. Based on this session, the proxy can add data representing the session ID to each subsequently derived file handle. If the session expires under certain conditions, the client must authenticate again to access any data from the file server.

Figure 1 is a block schematic diagram that shows how a proxy appliance or software program can distribute information in client file handles according to the invention.

The presently preferred embodiment of the invention uses file handles sent to clients 10 by a proxy 12 to store information that is relevant to those file handles and the possible operations that can be executed with them. In this way, even long after the proxy has lost track of a specific file handle, *e.g.* because of limited memory, whenever the client that has the file handle requests an operation with it, for example, to WRITE a block of data to a file server 14, the necessary metadata are immediately available to the proxy from that same client file handle.

As shown Figure 1, the client requests a file <filename> from the file server via the proxy using a LOOKUP call:<filename> which is passed via the proxy to the file server. The file server returns a LOOKUP reply:<file handle> to the proxy. The invention herein disclosed provides a mechanism by which the proxy inserts <metadata> into the file server's response to the client request and returns a LOOKUP reply:<<file handle>,<metadata> to the client. For example, in NFS version 3, if the file server returns a file handle of length 32 bytes, there are 32 more bytes available for inserting metadata. The file handle data from the server can be appended to that metadata, and the file handle size can be set to the sum of the length of the server file handle and the length of the proxy metadata. Alternatively, the client file handle can be obtained by combining the server file handle and the proxy metadata in some other way possibly, but not necessarily, involving cryptographic and/or compression transforms; or the file server file handle can be replaced completely.

Figure 2 is a block schematic diagram that shows how a proxy appliance or software program uses the information in client file handles, *e.g.* WRITE:call<<file handle>,<metadata>, when it receives a request from a client according to the

invention. The real file handle that is sent to the server then is the client file handle <file handle> without the metadata <metadata> that is only intelligible to the proxy. The metadata <metadata> serves its purpose by making it possible to handle the client request appropriately on the proxy, for example, by using the appropriate cryptographic keys to transform the data in the request or reply, or even by helping determine which server the request is ultimately intended for in the case of virtualization.

Figure 3 is block schematic diagram that shows the additional operations that must be executed if file identity information is missing from the file handle. It can be seen when contrasting the example of Figure 3, which shows the state of the art, with that of Figure 2 that the WRITE call:<file handle> operation is streamlined when using the approach disclosed herein (Figure 2) because file identity information is contained within the <metadata>, which is a part of the file handle. However, the state of the art approach requires an additional exchange between the proxy and the file server, *i.e.* READ:call<file handle> and READ reply:<metadata>, to establish file identity before the WRITE call:<file handle> operation may be performed.

Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention.

For example, invention may be practiced with any protocols that allocate a handle or, in general, an identifier to each new storage object, *e.g.* a file, and require that further accesses to this object include this handle/ID instead of the name. Such

protocols may be stateless or statefull. Further, the handle may be modified or supplanted entirely by metadata.

Accordingly, the invention should only be limited by the Claims included below.

5